

**PATENT**  
**47079-00191**

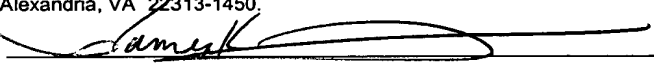
**APPLICATION FOR UNITED STATES LETTERS PATENT**

**for**

**GAMING MACHINE HAVING TARGETED  
RUN-TIME SOFTWARE AUTHENTICATION**

**by**

**Chad A. Ryan**

EXPRESS MAIL MAILING LABEL	
EXPRESS MAIL NO.:	EV 306223137 US
DATE OF DEPOSIT:	July 9, 2003
I hereby certify that this paper or fee is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.	
Signature:	

**GAMING MACHINE HAVING TARGETED RUN-TIME  
SOFTWARE AUTHENTICATION**

**REFERENCE TO RELATED APPLICATIONS**

5           This application is related to U.S. Patent Application Serial No. 10/119,663  
filed on April 10, 2002, entitled "Gaming Software Authentication", and incorporated  
herein by reference in its entirety.

**FIELD OF THE INVENTION**

10           The present invention relates generally to gaming machines, and more  
particularly, to software authentication of programs running in a gaming machine.

**BACKGROUND OF THE INVENTION**

15           As a regulatory requirement in virtually all jurisdictions that allow gaming, it  
is necessary to have a technique to authenticate that the software installed in a gaming  
machine is tested and approved. In the past, gaming manufacturers have generally  
used EPROM-based hardware platforms to store program code. As a result, a number  
of software authentication techniques have been accepted as standards throughout the  
gaming industry. Depending upon the preferences of the local regulatory agency,  
20       these techniques generally include either a Kobetron signature or a hash function  
based on the data stored in the EPROM device.

          Authentication of software programs basically occurs using two different  
methods in the field, again determined by the local regulatory agency. In one method,  
each EPROM is authenticated by a gaming agent prior to being installed in a gaming  
25       machine that is to be brought up for play. The EPROMs may be shipped directly to  
the gaming agency for authentication prior to the install date of the machine, or may  
be authenticated on the casino floor as the software is being installed in the machine.  
In another method, authentication is conducted on a spot-check basis wherein a  
gaming agent periodically visits a casino and picks machines for the removal of  
30       software components for authentication.

          Jurisdictional requirements require that storage media containing code or data  
be authenticated at power-up, continuously or at a periodic rate, or upon occurrence of  
predetermined events, such as the opening any doors or panels of the gaming device  
that allows access to internal circuitry. The storage media may be comprised of

erasable programmable read-only memory devices (EPROMs), electrically erasable programmable read-only memory devices (EEPROMs), PROMs, CompactFlash storage cards, hard disk drives, CD drives, or substantially any non-volatile memory and in some cases volatile memory (e.g., NVRAM, specialty mask semiconductors, battery backed RAM, SRAM, DRAM, etc.). Storage media comprises a memory device and the data stored thereon. Authentication of storage media is controlled by the gaming device's central processing unit (CPU). However, authentication by the CPU may take more than several minutes due to increasing complexity of the gaming device's software and thus the storage size of the media.

For example, the authenticity of numerous storage devices associated with the CPU may need to be determined every so often while a gaming machine is running. In some cases, gaming authorities require that a gaming program be authenticated about every ten minutes while the gaming machine is running. To determine the authenticity of a memory device's contents the CPU must read the memory device and perform various calculations and comparisons to determine if the memory device's contents are authentic. Reading many memory devices or large memory devices can use significant CPU time and therefore may negatively affect the responsiveness of the gaming program that a user interacts with. What is needed is a technique for authenticating memory devices associated with a gaming machine that does not affect the gaming program that the user interacts with.

## SUMMARY OF THE INVENTION

Embodiments of the present invention authenticate a gaming machine program, software, firmware, or data (data) stored in memory devices within the gaming machine while the gaming machine is running and interacting with a user. The authentication process does not slow or interfere with the gaming program that interacts with the user. The authentication processes are ongoing and are substantially continuously repetitive. The authentication processes may substantially repeat every 2 minutes to 24 hours. Furthermore, in order to increase the speed of authenticating some of the data, graphics data may be differentiated from executable data so that the authenticity of the executable data can be determined more often than the graphics data.

It should be understood that for the purposes of this description of exemplary embodiments of the present invention that the gaming machine program may be either compiled or uncompiled. Furthermore the gaming machine program comprises code, files, instructions or programs that are executable in that they direct the gaming machine to do something (hereinafter “executable code”). The gaming machine program further comprises graphics code, files, data, instructions or programs (hereinafter “graphics data”) that have to do with graphics or multimedia applications. Such graphics or multimedia may include, but are not limited to, data or information used to control graphics, animation, or other special effects that move air, move fluids, create smells, create bubbles, create flashing lights, control laser lights, control air pressure, control temperature, control mechanical devices, or control sound devices.

In an embodiment of the present invention a gaming machine comprises a user interface and a central processing unit (CPU) coupled to the user interface. The CPU comprises a processor. A first memory is coupled to the processor. The first memory is adapted to contain executable program code. The executable program code has both executable instructions and graphics data. A second memory is also coupled to the processor. The second memory stores data. The executable instructions found in the first memory include a plurality of instructions configured to cause the processor to determine the authenticity of the executable program code and the data. The processor, with the aid of the executable instructions, determines the authenticity of the executable program and the data on a substantially continuous, repetitious basis. Furthermore, the authenticity determination of the executable program code might be performed substantially in a parallel process with the authenticity determination of the data.

Other executable instructions cause the processor to determine, when reading said executable program code, whether executable code or graphics data are being read. If the processor is reading graphics data, then the plurality of executable code cause said processor to not determine the authenticity of the graphics data unless more than a predetermined number of events have passed since the last time the graphics data was authenticated. The events may be seconds, clock count-ups, countdowns, a number of repetitions through a program loop, etc. If the processor reads executable instructions, then the plurality of instructions cause the processor to determine the authenticity of said executable instructions.

In another embodiment of the present invention, a gaming machine comprises a CPU and a plurality of memory devices. The CPU is adapted to determine the authenticity of data in at least two of the plurality of memory devices in a parallel, repeating manner. The CPU is also adapted to read the data stored in at least one of the plurality of memory devices and determine whether the data is executable data or graphics data. If the data stored in a memory is determined to be graphics data, then the CPU is adapted to determine the authenticity of the graphics data if a predetermined number of events have passed. The events may be clock cycles, seconds, count-ups, count-downs, passes through a program routine or loop, uses by a user, number of games played, etc. If the data stored in a memory is determined to be executable data, then said CPU is adapted to determine the authenticity of said executable data.

In another embodiment of the present invention a method is provided for authenticating various memory devices' data within a gaming machine while the gaming machine is operating. The authentication process of the memory devices' data can be performed in substantially a parallel fashion, such that two or more memory device's data is being authenticated at about the same time. The method of authenticating also comprises reading data from a first memory device and determining or using other techniques to determine whether the data is graphic data or executable code. If the data is determined to be graphic data, then more data is read from the first memory device. If the data is determined to be executable data, then it is determined whether the executable code is authentic after which more data is read from the first memory device.

The above summary of the present invention is not intended to represent each embodiment, or every aspect, of the present invention. This is the purpose of the figures and the detailed description which follow.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

The foregoing and other advantages of the invention will become apparent upon reading the following detailed description and upon reference to the drawings.

Figure 1 is an isometric view of a gaming machine operable to conduct a wagering game;

Figure 2 is an exemplary block diagram of a CPU in a gaming machine according to the present invention; and

Figure 3 is a flow chart for an exemplary run-time authentication process for a gaming machine.

While the invention is susceptible to various modifications and alternative forms, specific embodiments have been shown by way of example in the drawings and will be described in detail herein. It should be understood, however, that the invention is not intended to be limited to the particular forms disclosed. Rather, the invention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the appended claims.

## **Description of Illustrative Embodiments**

Turning now to the drawings and referring initially to Figure 1, a gaming machine 10 is operable to conduct a wagering game such as mechanical or video slots, poker, keno, bingo, or blackjack. If based in video, the gaming machine 10 includes a video display 12 such as a cathode ray tube (CRT), liquid crystal display (LCD), plasma, or other type of visual display known in the art. A touch screen preferably overlies the display 12. In the illustrated embodiment, the gaming machine 10 is an “upright” version in which the display 12 is oriented vertically relative to a player. Alternatively, the gaming machine may be a “slant-top” version in which the display 12 is slanted at about a thirty-degree angle toward the player.

The gaming machine 10 includes a plurality of possible credit receiving mechanisms 14 for receiving credits to be used for placing wagers in the game. The credit receiving mechanisms 14 may, for example, include a coin acceptor, a bill acceptor, a ticket reader, and a card reader. The bill acceptor and the ticket reader may be combined into a single unit. The card reader may, for example, accept magnetic cards and smart (chips) cards coded with money or designating an account containing money.

The gaming machine 10 includes a user interface comprising a plurality of push-buttons 16, the above-noted touch screen, and other possible devices. The plurality of push-buttons 16 may, for example, include one or more “bet” buttons for wagering, a “play” button for commencing play, a “collect” button for cashing out, a “help” button for viewing a help screen, a “pay table” button for viewing the pay table(s), and a “call attendant” button for calling an attendant. Additional game-specific buttons may be provided to facilitate play of the specific game executed on the machine. The touch screen may define touch keys for implementing many of the

same functions as the push-buttons. Other possible user interface devices include a keyboard and a pointing device such as a mouse or trackball.

Referring now to Figure 2, a central processing unit (CPU) 30 controls operation of the gaming machine 10. In response to receiving a wager and a command to initiate play, the CPU 30 randomly selects a game outcome from a plurality of possible outcomes and causes the display 12, via the video circuitry 39 and video out 40, to depict indicia representative of the selected game outcome. Alternatively, the game outcome may be centrally determined at a remote computer using either a random number generator (RNG) or pooling schema. In the case of slots, for example, mechanical or simulated slot reels are rotated and stopped to place symbols on the reels in visual association with one or more pay lines. If the selected outcome is one of the winning outcomes defined by a pay table, the CPU 30 awards the player with a number of credits associated with the winning outcome.

The CPU 30 includes a microprocessor 32 and various memory devices (media devices). The microprocessor 32 interfaces with many other components of the gaming machine 10 via an interface bus 34. A main memory 36 stores the compiled gaming machine program for operating the gaming machine 10.

The main memory 36 may be DRAM or SRAM or substantially any other volatile memory device or reprogrammable non-volatile memory device. The battery backed memory 38 stores machine critical data that cannot be lost when power is removed from machine 10. The battery backed memory 38 may be battery backed volatile memory or a reprogrammable or rewritable non-volatile memory device. The video circuitry 39 supplies display information to a video display 12 that may comprise a CRT, LCD, plasma, or other display device. Audio circuitry 42 generates sounds for game play on the gaming machine 10. The I/O control 44 controls input/output interfaces with the user interfaces such as game buttons 16, coin validators 14, touch screen bill validators, multimedia devices, etc.

In an exemplary embodiment, the various memory devices may also include a boot memory 46, a high capacity storage memory 48, and a serial read-write memory 50. The boot memory 46 is preferably a read-only memory such as a one megabit EPROM, EEPROM, PROM or other type or appropriately sized programmable read-only memory. The boot memory 46 may also be substantially any type of non-volatile memory. The high capacity storage memory 48 is preferably a CompactFlash card, but may also be a hard disk drive, CD drive, DVD drive, magnetic RAM, battery

backed RAM or other type of non-volatile memory. The serial memory 50 is preferably an EEPROM such as a 512 byte SPI EEPROM, but could be any type of programmable read-only or read/write non-volatile memory. Depending upon the preferences of the local gaming regulatory agency, all three memories may be adapted  
5 to be authenticated outside of the CPU as well as when installed in the CPU at power up or prior to being utilized in the gaming machine.

The boot memory 46 stores, at least some of the following, boot code 52, an authentication program 54, a RAM loader, a decompression utility 56, and a digital signature 58. The authentication program includes a hash function 60, a digital  
10 signature verification algorithm 62, and a public key 64. The hash function 60 may, for example, be an SHA-1 hash algorithm that reduces a data set to a unique 160 bit message digest. A hash algorithm or function is used to calculate a message digest corresponding to the files in, for example, a memory device. The message digest does not have to be unique, i.e., the function may return the same hash value for two or  
15 more items (although this is very unlikely). The non-uniqueness of the hash value for each item in the message digest is acceptable because each hash value is used to evaluate a different file or data set within a memory device. The message digest is a small representation of a large amount of data. A message digest is a relatively unique representation of data, from a cryptographic standpoint, and is an irreversible  
20 representation of the data. In other words, one cannot recreate the original data from the message digest.

The digital signature 58 is generated, in effect, from the boot memory's contents as a whole. In an exemplary embodiment, after hashing is performed to produce a message digest, then a digital signature is created to enable the origin and  
25 authenticity of the digest to be determined. When there is data that requires a means for determining the origin of the data, one generally uses a digital signature mechanism. There exists a federal standard called FIPS 186-2 that defines a digital signature generation and verification mechanism called the Digital Signature Algorithm (DSA). In an exemplary embodiment a digital signature is created from  
30 the message digest. In essence the DSA uses a private key, a public key, and the message digest. A private key and the message digest are used to create an original signature associated with the original message digest. The public key, the original signature, and a calculated message digest are used to check a signature associated with a message digest in order to determine the origin and authenticity of the data set.



It is understood that neither the message digest nor the data or files used to create the message digest can be recreated using the DSA. The digital signature 58 is used to sign the message digest of the boot memory contents. Again, the signature may be used to determine the source or manufacturer of the message digest, via a public key, but cannot be used to recreate the message digest or the original data. Furthermore, the DSA is not being used here as an encryption process under FIPS 186-2, but rather a technique for validating the signature associated with the data set, and the public key.

The high capacity storage memory 48 stores game and operating system executable program files 66, sound operating system files 68, sound bank files 70, graphics files 72, a file list of file types 74, and digital signatures 76, 78. The files in the high capacity storage memory 48, taken together, constitute a “gaming program” as that term is used herein, and the various files constitute “data files” as that term is used herein. Thus, the gaming program includes a plurality of data files. For each data file on the high capacity storage memory 48, the manifest file contains a file name, a file type, a load address, and a file digital signature 76. The whole device digital signature 78 is generated from the gaming program as a whole, while each digital signature 76 is generated from the associated data file listed in the manifest file.

The serial read-write memory 50 stores information/data specific to the jurisdiction where the CPU is to be installed. This information may, for example, include a lottery terminal identification (ID) 80, a part number 82, a jurisdiction ID 84, a jurisdiction name 86, jurisdiction bit code options 88, jurisdiction max bet 90, jurisdiction max win 92, and a digital signature 94. The digital signature 94 is generated from the serial memory’s contents as a whole.

The boot memory 46, serial read-write memory 50 and high capacity storage memory 48 may each be removable devices and/or contain alterable software. Each of these memory devices may be able to be reprogrammed or be able to receive downloaded updates from an outside source via a programming device, a network such as the Internet, an intranet, an Ethernet, a fibre loop, or other type of networking system. The boot memory 46, serial read-write memory 50, and high capacity memory 48 each may be required to be authenticated by the gaming machine 10 at various points during operation of the gaming machine.

In order to better understand the advantages of an exemplary run-time authentication algorithm, it is important to realize that as gaming machines evolved they began to use alterable media, such as flash memories, EEPROMs, EPROMs, CD drives, disk drives, etc. in their electronics and programming structure to store all or portions of the programs and files. Newer gaming machines are designed to allow the gaming software to be updated, to grow in size, and to grow in complexity. Because of these advances and changes in gaming machine design, electronics, software and memory storage size the time necessary to authenticate the software in the storage media during run-time operations has increased because the methods required to authenticate the software content became more complex. An increase in the time required to authenticate the software during machine run-time operations may affect the responsiveness and speed of the run-time software as well as the smoothness of operation to the extent that it is noticeable to the user. A CPU may become unable to effectively operate the gaming machine main program while multiplexing authentication processes are taking place due to the sheer size of the main program that must be authenticated within a predefined period of time. Thus, it is necessary to provide a technique to authenticate the gaming programs and media within various media devices without slowing or disturbing the operation of the gaming machine.

An exemplary run-time authentication comprises two main cycles of events during the operation of a gaming machine. The first cycle of events checks whether the high capacity storage memory 48 is connected to the bus 34. This check is performed at predetermined intervals that may range from about every 5 ms to about every minute. The first cycle also checks whether the high capacity storage memory's 48 SHA-1 message digest calculation is continuously being recalculated.

The second cycle of events performs a constant or continuous authentication of the boot memory 46, the serial read-write memory 50, the files that are being executed from the main memory 36, and the integrity of the data stored in the battery backed memory 38. Utilizing a SHA-1 hash message digest of a media device's contents the authentication of each media device is performed. The authentication of the media device during a run-time authentication may be limited to the data in the whole media device rather than the individual files stored in the media device. The authentication of a media device may also be performed file by file when the CPU has stored the memory locations and the type of data in the memory locations prior to an authentication process.

During a boot-up process of the CPU 30 the media devices and software thereon are normally authenticated. The boot-up authentication process includes performing a SHA-1 hash over the media software that is loaded into the main memory 36, authenticating the digital signature 58, 78, 94, and storing the calculated hash message digest in battery backed memory. Thus, during run-time authentication there is no requirement to perform signature verification since the files and components were proven to be authentic during the boot process. One main purpose of run-time authentication is so the CPU 30 can check to make sure that the files and data loaded into the main memory 36 during the boot process have not been altered. Another purpose of the run-time authentication is to verify that certain software or hardware components, such as the boot memory 46, the high capacity storage memory 48, or the serial read-write memory 50 have not been changed or undergone a change in any of their software/firmware. In order to check the executable code in main memory 36, the boot memory 46, the high capacity storage memory 48, or the serial read-write memory 50 for authenticity, only a SHA-1 hash, or its equivalent is necessary since all had been verified at boot-up to have come from a trusted source via a digital signature verification process. It is understood that there are various other techniques other than a SHA-1 hash function that could be used to verify the authenticity of the various media devices during run time. Such other techniques may include, but are not limited to, CRC-16, CRC-32, MD5 and checksum techniques.

As an additional run-time authentication and verification check, a digital signature verify operation is performed on the media devices (e.g., main memory 36, boot memory 46, high capacity storage memory 48, and serial read-write memory 50) when the gaming software returns from certain gaming events. These events are mainly security events wherein people have had access to the inside of the gaming machine or the gaming machine has made a large payout. The security events that may require an additional run-time verification and authentication check along with a digital signature verify operation include, but are not limited to:

1. Any "door closure event". On a gaming machine there may be various doors or hatches for providing access to the interior of the gaming machine. Anytime one of the doors or hatches is closed, the gaming program and other various media devices are checked for authenticity because someone may have had access to the interior of the gaming machine.

2. Any return to game play when exiting the “administration screen”. Various gaming machines have an administration mode. There may be one or more levels for the administration mode. For example, one mode may include critical configuration settings affecting the payouts made by the gaming device and may require machine doors or hatches to be accessed to gain entry. Another mode may allow an administrator to view and verify meters, event logs, game playtime, machine statistics and other items benign to the functionality of the gaming device without unlocking any machine access doors or hatches.
3. Any return to game play from a “game disable” state. An attendant, a command from a host system, or other internal mechanisms can place the gaming machine in a game disable state in order to reserve the gaming machine for a certain player or for numerous other reasons. Essentially the gaming machine is on, but will not operate until it is taken out of the disabled state.
4. Any cashout handpay state. A cashout handpay is typical when a player would like to cash out of gaming machine and the amount of credit or winnings on the gaming machine is higher than the amount of coins or payout units in the gaming machine’s hopper or higher than an operator configured machine payout limit. If this occurs, the gaming machine may go into a cashout handpay state wherein an attendant will have to come to the gaming machine and assist the player so that the player can get manually paid or handpaid. Once the cashout handpay is completed the attendant will use a key, card or other code or device to access the gaming machine and exit from the cashout handpay state.
5. Any Jackpot handpay state. A Jackpot handpay state is similar to the cashout handpay state, except the gaming machine is set to go into a Jackpot handpay state when a jackpot hit by the player is above a predetermined amount such as a monetary amount that must be reported to Internal Revenue Service (IRS). When a jackpot of the predetermined amount or greater is hit then the machine locks up and an attendant is called to hand pay the player and further to have the player fill out the appropriate IRS (W-2G) form(s). The attendant can then use a key, card, pass code, or other appropriate means to reset the gaming machine into a play mode again.

After a successful verification of all files in main memory 36, the battery backed memory 38 is verified using, for example, a CRC check. The battery backed memory 38 can be set to store two copies of critical data -- a first copy that is stored as a master copy and a second copy that is stored as an auxiliary copy. The master copy program and auxiliary copy of the critical data can also be compared to each other to help ensure the integrity of the critical data being stored in the battery backed memory 38.

Figure 3 depicts a flow chart of an exemplary authentication process for continuous run-time authentication in accordance with an embodiment of the present invention. After boot-up of the gaming machine, wherein gaming machine program software or firmware was authenticated in at least one of a variety of accepted ways, and while the gaming machine is operational, the CPU 30 will, in conjunction with executing the gaming machine program, continuously authenticate the main memory 36, battery backed memory 38, boot memory 46, high capacity storage memory 48, the serial read-write memory 50 and any other memories that may require authentication. The CPU can be set to authenticate substantially any media device in the gaming machine or closely associated with the gaming machine through a network. The main application is launched at step 100 from the main memory 36. The gaming machine is operational and the authentication of predetermined media devices begins. From step 100, two authentication functionalities operate substantially in parallel as depicted by path A and path B. Path A authenticates the high capacity storage memory 48, and path B authenticates, in a serial fashion, the main memory 36, the battery backed memory 38, boot memory 46, and the serial read-write memory 50. The dotted line for path C indicates that other authentication processes may also take place in parallel with path A and B.

Discussing path A first, a predetermined amount of data is read from the high capacity storage memory 48 at step 102. Path A is separated from path B because the high capacity storage memory 48 may include a much larger amount of data than that which is found on path B. By separating the paths, all components on path B can be authenticated one or more times in the same time as one cycle through path A. The predetermined amount of data may be a bit, a byte, a word or, for example, 1 bit to 1 Kbytes of data, or any amount of data that the architecture can handle in the time allotted for the function. The CPU processes the gaming machine program and

performs the authentication functionalities in a time sharing manner. The percentage of sharing depends on how the sharing affects the gaming machine program's main application that interacts with a user while completing the authentication within a predetermined amount of time.

5           At step 102 the data that is read is used to calculate a hash message digest representative of the data. At step 103, the CPU determines whether all the data in the high capacity memory 48 has been read in order to determine if the hash calculation is complete. If all the data from the high capacity memory 48 has not been read then the algorithm returns to step 102 to read more data and continue calculating the hash  
10   message digest. If at step 103 the hash calculation for all the data has been completed then, at step 104, the calculated hash message digest is compared with a previously stored hash message digest result for the data contents of the high capacity storage memory. The stored hash result may have been stored in one of the various non-volatile memories in the gaming machine. For example, the stored hash result may  
15   have been stored in a battery backed NVRAM 38 during boot-up. If the verification comparison indicates that the calculated hash message digest and the stored hash message digest are the same, then the high capacity memory is considered authenticated and the algorithm returns to step 102 and begins reading data from the high capacity storage memory 48 from the beginning (or any predetermined data  
20   location) again. This loop continues for as long as the gaming machine is powered on. If the verification comparison fails, at step 104, due to the stored hash not being equal to the calculated hash, then a critical error is displayed, at step 105, on the gaming machine. The gaming machine then becomes non-functional or out-of-order until an attendant comes over the machine and determines what needs to be done to  
25   correct the error.

          Ideally, the high capacity storage memory has the predetermined amount of data read from it about every 15 ms, but the data reading loop of path A may be substantially any amount of time, for example, from between 2 ms to once a day so long as the read takes place within the limitations of CPU. It is understood that in an  
30   exemplary embodiment of the present invention, the high capacity storage memory 48 is not the device from which code is executed. For example, the high capacity memory 48 may be a compact flash card, a hard drive or other type of non-volatile memory device that cannot be used to execute the gaming program. In many circumstances, the high capacity memory 48 may be hot-plugable or hot-swappable

with the gaming machine. As such, the run-time validation of the high capacity memory 48 also functions in various ways, as a check or means for making sure the high capacity memory has not been removed, unplugged or partially disconnected from the gaming machine after boot-up.

5           Furthermore, the high capacity memory 48 may be a non-volatile memory capable of providing an executable program to the microprocessor 32. If this is so, an exemplary embodiment of the invention may not be required to have both a main memory 36 and a high capacity memory 48.

10           It should be noted again with respect to path A, that there might be more than one high capacity memory that must be authenticated. Path C (dotted line) represents an algorithm wherein one or more additional high capacity memories (or other media devices) are part of the CPU 30 in an exemplary gaming machine. The data in the additional memories may be authenticated via similar means and in parallel with paths A and B.

15           With respect to path B coming out of step 100, at step 106 data is read from the serial read-write memory 50 and a hash message digest is calculated from all the bits. In the exemplary embodiment the serial read-write memory 50 contains significantly less data than the high capacity memory 48. Since there is significantly less data in the serial read-write memory 50 than the high capacity memory 48, the  
20           data in the entire memory can be read as a binary image such that a hash calculation can be performed. The hash calculation result is then compared with a stored serial read-write memory hash message digest that was calculated at boot-up. If the two hash message digests do not match, then the algorithm indicates that the authentication failed and a critical error is displayed on the gaming machine at step 5.  
25           On the other hand, if the stored and calculated hash message digests match, then the serial read-write memory contents are considered validated and authentic.

          At step 107, the boot memory's data is read and a hash message digest is calculated. The calculated boot memory hash is compared with a boot memory hash message digest that was stored at boot-up. If the hash message digests do not match,  
30           the fail path is taken to step 105 and a critical error is displayed on the gaming machine. If the hash message digests match then the boot memory data is validated. Another step could be placed here to validate any other memory associated with the CPU 30. These additional steps may be substantially the same as steps 106 and 107. Once steps 106 and 107 (and any other similar steps) are completed the algorithm

goes to step 108. Either path A or B may have one or more authentication processes performed in a serial fashion.

In an exemplary embodiment of the present invention the main memory 36, or other memories (such as the battery backed memory 28 or possibly the high capacity  
5 memory 48) may contain both executable code along with graphics data. Executable code and graphics data may be compiled code or uncompiled code. When the gaming machine program (the game executable, operating system executable, and all graphics data) is compiled as a single compiled gaming machine program and stored in the main memory 36 (or other memories) the single compiled gaming machine program  
10 can be quite large and take a significant amount of time to authenticate when compared to the time required to authenticate, for example, the boot memory 46. Table 1 illustrates approximate authentication times of compiled or executable programs or files an embodiment of the present invention.



Executable Program Size	Average Verification Time
1.5 MB	1.9 minutes
3.0 MB	3.8 minutes
4.5 MB	5.7 minutes
6.0 MB	7.6 minutes
7.5 MB	9.5 minutes
9.0 MB	11.4 minutes

**Table 1**

If a first gaming machine has a gaming machine program in its main memory that is about 1.5 MB, then the authentication time is within a reasonable time frame of less than about 10 minutes. If a second gaming machine has a gaming machine program in its main memory that is greater than about 6.0 MB, then the time required to authenticate begins to become unacceptable due to gaming agencies requesting that the gaming software be authenticated about every 10 minutes while the gaming machine is powered on.

Assume that the main difference between the first and the second gaming machine is that there is more graphics data in the second gaming machine's gaming machine program. Then, it is understandable that if the compiled executable code in both gaming machines are about the same size (give or take a few megabytes), then a separation of executable code portions of the gaming machine program from the graphics data portions would decrease the time required to authenticate the second machine's compiled gaming machine program to about the same amount of time as the first machine's compiled gaming machine program. Furthermore, tampering with the executable data may be more harmful to a user of the gaming machine than tampering with the graphics data. This is because adjusting or tampering with the executable part of the program may affect the proper odds and payouts of the gaming machine. Wherein tampering with the graphics data may have the lesser effect of disturbing the gaming graphics or other multimedia experience. As such, in one embodiment of the present invention the graphics data is separated and left out of the authentication cycle. This may be acceptable because all the graphics data is called by the executable code, which is constantly authenticated. In another embodiment of the present invention, the graphics data is authenticated on a less frequent basis in

order to offload the processor so that more time can be dedicated to authenticating the executable code files. For example, the graphics data in the main memory may only be authenticated from every other time the executable code is authenticated to once every hour, day, at predetermined intervals, or after a predetermined number of events. A timer or counter may be utilized to measure a predetermined number of events such as clock counts, cycles, up-counts, down counts, seconds, number of games played, number of users, etc.

Still looking at step 108 of Figure 3, an exemplary authentication algorithm begins to read data from the main memory (SDRAM) 36 and determined if the data being read is executable code or some another type of code such as graphics data. The determination of whether data is graphics data or executable code can be made prior to reading the data. Reading data and the determining whether the data is graphics data or executable code can take more time than already having loaded static memory addresses of indicating whether data in such static memory addresses is graphics data or executable code. As such, in embodiments of the present invention, static memory addresses are loaded into one of the volatile or non-volatile memories indicating where all the data is, for example, in the main memory 36 or the high capacity memory 28 and what type of data it is. In other exemplary embodiments of the present invention wherein data is dynamically loaded into a memory device, various exemplary methods can be utilized to identify the data locations and the data type. For example, a list indicating which memory locations are storing graphics data and which memory locations are storing executable code can be created and stored at the time the data is loaded into a memory device at boot-up during programming of the device, or any other time. Such a list allows the CPU to forego reading the data before making a type-of-data determination.

If the data read is executable code (e.g., belongs to an executable file), then at step 110 a hash calculation is performed on the content of that executable file. The hash message digest is compared against the hash message digest that was stored in a non-volatile RAM at, for example, boot-up for the particular executable file. If the hash message digests do not match, then authentication fails and a critical error is displayed at step 105. If the signatures match and are verified, then the executable file is authenticated. At step 112 it is determined whether all the executable files in the main memory have been read. If all the files have not been read then the algorithm goes back to step 108 to read the next file or predetermined amount of data.

If the next file or predetermined amount of data that is read at step 108 is not executable code, then at step 109 a timer or counter is checked. If this is the first time through the algorithm loop, then the algorithm will automatically go from step 109 to step 111 wherein non-executable data or files (e.g., graphics data or files) are authenticated. If this is not the first time through the algorithm loop, then the timer, counter or countdown (hereinafter “timer”) is checked to determine whether a predetermined amount of time (counts or number of events) have passed. If the predetermined amount of time had not passed, then the non-executable file is not authenticated at step 111 and the algorithm returns to step 108 to read the next file. If the timer has reached the predetermined amount of time (counts), then the graphics file is checked for authenticity via, for example, by comparing a calculated hash message digest with a previously stored hash message digest as discussed previously. If the non-executable file cannot be authenticated by the calculate-and-compare hashes method, then a critical error is displayed at step 105. If the non-executable file is authenticated by calculating a hash message digest and then comparing the calculated message digest with a stored message digest for the file, then the algorithm checks whether the last file in the main memory has been read at step 112. If the last file has not been read, then the algorithm returns to step 108 and reads the next file or predetermined amount of data. If the last file has been read from the main memory 36 at step 112, then the battery backed memory 38 is checked at step 113.

With respect to step 109, another exemplary embodiment may have a timer for each of the graphics data files so that the more critical graphics data files can be set to be checked more often than, for example, a non-critical graphics data file. Another exemplary reason for giving each graphics data file its own timer would be to stagger the authentication of the non-executable files in order to limit loading on the microprocessor 32.

The battery backed memory 38 is checked at step 113. In an exemplary embodiment a cyclic redundancy check (CRC) is performed on the nonvolatile RAM, battery backed memory 38. A CRC is a technique for detecting data changes or errors. A checksum or perhaps a hash calculation could also be used to authenticate the battery backed memory 38.

If the battery backed memory 38 is not determined to be authentic, then a critical error is displayed at step 105. If the battery backed memory 38 is authenticated, then the exemplary algorithm checks to make sure the machine is

running at step 114. If the machine continues to be operational then the loop returns to step 106 wherein the authentication of data within the selected memory devices is repeated in a serial manner and substantially in parallel with the authentication of the high capacity memory 48.

5           While the present invention has been described with reference to one or more particular embodiments, those skilled in the art will recognize that many changes may be made thereto without departing from the spirit and scope of the present invention. Each of these embodiments and obvious variations thereof is contemplated as falling within the spirit and scope of the claimed invention, which is set forth in the following  
10       claims.